# Programmation Web « Client Riche »
# M413 - TD n°5 (séance 7)

## 1 Objectifs

Ce TD illustre la partie du cours sur la programmation web et en particulier les concepts liés à la conception d'application riches coté client, c'est-à-dire dans votre navigateur.

Les principaux concepts abordés seront :

- Les principes avancés du langage JavaScript.
- L'exploitation du DOM.
- L'Ansychronisme en JavaScript : AJAX, XML et JSON.
- Mise en œuvre d'une application en JavaScript.

Vos « **livrables** » qui seront à déposer dans **la boîte de dépôt ad-hoc sur le LMS Moodle d'UCA avant la date spécifiée**, seront basés sur l'arborescence fournie et constitués **a minima** des **dossiers assets/, css/, js/** et des **fichiers** suivants :

- Les réponses à chacune des questions posées devrons être rédigées au format « Markdown » dans le fichier « **README.md** » fournie.
- Des fichiers index.html, *.html, js/td{x}.js js/*.js, css/td{x}.css, css/*.css et plus si nécessaire…

https://fr.wikipedia.org/wiki/Markdown

## 2 Documentation

Pour ce TD, mais également pour les suivants, sauf mention contraire dans l'énoncé ou si un lien vous est fourni, vous ne devez pas aller chercher des réponses sur internet !

**Seuls les sites suivants sont autorisés :**

- Le cours sur : https://lms.univ-cotedazur.fr/
- Le site de Mozilla : https://developer.mozilla.org/fr/
- Les sites du W3C : pour la validation HTML5 ou du CSS
  Mais aussi pour la documentation sur le DOM.

# 3 Prise en main de l'environnement

Pour réaliser ce TD, vous aurez également besoin d'un éditeur de texte pour écrire le code de vos pages. Vous pouvez par exemple utiliser un des logiciels **Brackets**, **Notepad**++, **Eclipse**, etc.

Pensez à sauvegarder régulièrement votre travail et à commenter votre code.

N'hésitez pas à sauvegarder les différentes versions, évolutions d'un même exercice de manière à pouvoir facilement réviser ou le réutiliser par la suite…

# 4 Les Outils pour parcourir le Document Object Model

De nos jours, des outils de débogage évolués sont intégrés aux différents navigateurs ( cf. cours). Ces outils vous seront très utiles pour explorer l'arbre DOM de votre page, vérifier les propriétés CSS ou encore connaitre les erreurs de votre code JavaScript !

# 5 Rappel de cours.

Lors de ce TD, l'ensemble des pages web écrites devra (sauf mention contraire) être constitué de pages XHTML5 valide ne contenant pas de mise en forme ( squelette index.html et *.html fournis).

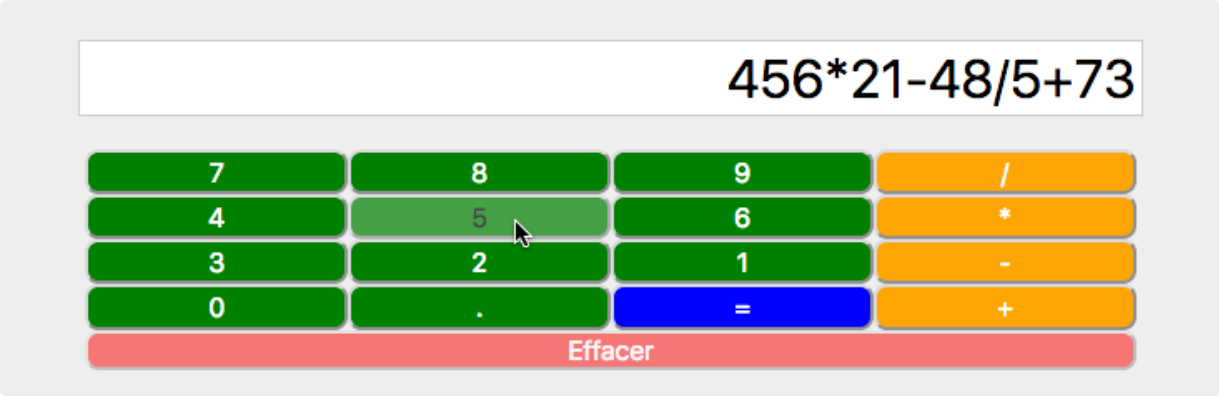
La mise en forme sera dans un ou plusieurs fichiers CSS valides
( squelettes css/td{x}.css css/*.css fournis).

De même, le code JavaScript sera également dans des fichiers séparés
( i.e. squelettes js/td{x}.js js/*.js fournis).

*Bien lire l'ensemble des consignes avant de commencer à coder.*

# 6 Exercice 1 : « Basic Calc » (obligatoire).

vous devez programmer une application Javascript qui génère une calculatrice basique dont l'aspect devra être le suivant :




Pour cela, vous vous baserez sur les différents squelettes index.html, css/td5.css et js/td5.js fournis.

Le fichier index.html fourni ne doit pas être modifié, seuls les fichiers css/td5.css et js/td5.js peuvent l'être.

Votre calculatrice devra être « contenue » dans la  « my-basic-calc ».

Le cahier des charges est le suivant :

- Les dimensions seront exprimées uniquement en % ;
- Les « font-size » seront exprimées uniquement en « em » ;
- La zone d'affichage des opérations et du résultat (en haut) : élément « input », largeur 90%, « font-size » : 2em, class="result". Sa valeur initiale est 0 ;
- Les boutons (nombres, opérateurs, =) : élément « input », « font-size » : 1em, lors du survol : opacité 0.7 et couleur de police noire sinon opacité 1.0 et couleur de police blanche. Les boutons nombres seront de couleur verte, ceux des opérateurs de couleur orange et celui du « = » de couleur bleu. Ils auront comme classe CSS respectives : class="btn number", class="btn operator" et class="btn equal". Des « clicks » succésifs sur les nombres et les opérateurs (i.e. /, *, -, +) devront permettre de saisir le(s) opération(s) à effectuer. Enfin, un « click » sur le bouton « = » effectuera l'opération affichée ;
- Les boutons ci-dessus seront disposés dans une « div » dont l'id sera celui de la « div container » suffixer par la chaine de caractères "-btns". Elle aura comme classe CSS : class="btns" ;
- Le bouton « Effacer » : élément « input », « font-size » : 1em, opacité : 1, lors du survol : opacité 0.5. Il sera de couleur rouge. Il aura comme classe CSS : class="clear". Un clic sur ce bouton met à 0 le résultat dans sa zone d'affichage.

Pour effectuer l'opération, l'utilisation de la fonction « eval() » est dans un premier temps l'approche la plus simple :

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/eval.

Attention toutefois à utiliser la fonction « eval() » à bon escient car c'est une fonction dangereuse du point de vue de la sécurité.

Ne pas oublier de commenter votre code source !

Pensez à nommer vos variables convenablement (en anglais de préférence), commenter (en anglais de préférence) et bien indenter votre code.

L'ensemble des constantes devra être regroupé en début de fichier pour être facilement modifié en cas de besoin. On ne retrouvera donc aucune valeur de constante dans le code…

La qualité du code produit sera évaluée.

# 7 Exercice 2 : « Basic Calc » améliorée (obligatoire).

Afin d'améliorer l'expérience de l'utilisateur vous aller maintenant devoir lui permettre de saisir l'opération à effectuer à l'aide du clavier en sus de l'utilisation de la sourie.

https://developer.mozilla.org/fr/docs/Web/API/KeyboardEvent/key

https://developer.mozilla.org/fr/docs/Web/API/KeyboardEvent/key/Key_Values#numeric_keypad_keys

Les touches « = » ou « enter » permettrons d'effectuer l'opération affichée alors que la touche « C » (majuscule) mettra à 0 le résultat dans sa zone d'affichage.

Enfin concernant les problèmes de sécurité liés à l'utilisation de la fonction « eval() », une alternative simple consiste à utiliser l'objet global **Function**.

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Function

On peut ainsi créer une fonction de portée globale à partir d'une chaîne de caractère. De cette manière, vous pouvez écrire un analyseur personnalisé pour évaluer le code de la chaîne de caractère. Il est alors moins probable que les attaques possibles fassent des dégats par rapport à eval()…

```
// Safe alternative to eval() function
function parse(str) {
      return Function(`'use strict'; return (${str})`)()
}

// An arithmetic operation
let operation = '456*21-48/5+73';

// Get the result
let result = parse(operation);

// Log the result
console.log(result); // Will display 9639.4
```